Universitat de Lleida

TREBALL FINAL DE GRAU

ESCOLA
POLITÈCNICA SUPERIOR
UNIVERSITAT DE LLEIDA
INSPIRING THE FUTURE

Estudiant: Eduard Arnedo Hidalgo

Titulació: Grau en Enginyeria Informàtica

Títol de Treball Final de Grau: **MushroomApp: a Mushroom Mobile App**

Director/a: **Francesc Solsona Tehàs i Sergio de Miguel Magaña**

Presentació

Mes: Setembre

Any: 2019

MushroomApp: a Mushroom Mobile App

Author: Eduard Arnedo Hidalgo
Directors: Francesc Solsona Tehàs and Sergio de Miguel Magaña

Abstract

Background. Taking into account the mycological production of pine forests in Catalonia, more than 700 different species of mushrooms have been properly tagged and stored in a Data Base (DB). In this project we present *MushroomApp*. This App identifies mushrooms, by a simple image, from a corpus made up by the images of the DB. Supervised machine learning classifiers is an efficient mean for identifying mushrooms, and more specifically Artificial Neural Networks (ANN), so it was the one selected in this project. ANN models are created with Google Libray TensorFlow, positioned as the leading tool in the Deep Learning sector.

Objective. The objective is to be able to create efficient ANN models using TensorFlow. In addition, we want to investigate a machine learning system to gradually improve our models.

Methods. As there are many types of mushrooms, an important design decision was to mark the range of mushroms within the scope of the MushroomApp model. To implement the server we have used Python together with Django. The server is responsible for carrying out the operations of inserting new mushrooms and creating the TensorFlow models of the ANN. We will create these Models through Keras, a library that runs TensorFlow operations. The App is developed with Flutter to run the App on iOS and Android. Among its most important operations there are consulting the catalog, uploading images and making predictions.

Results. The Precision, Recall and F-score of the ANN models for genus detection have been obtained with a corpus of 10,000 and 27,000 images. Mushroom detection performance has also been measured with 7,000 and 5,000 images. The best F-score obtained has been less than 0.5.

Conclusions. The results obtained suggest an improvement and expansion of the corpus to increase the performance of the models obtained.

Index Terms

App, Mushroom, Android, iOS, Python, Django, Flutter, TensorFlow, Keras

CONTENTS

LIST OF FIGURES

LIST OF TABLES

I. Introduction

Taking into account the mycological production of pine forests in Catalonia, we have been able to identify more than 700 different species of mushrooms. All of them have been stored in our database. We obtained these images from public sources, images without copyright on the Internet, and some of them are of our own. Many of these images have been treated and manipulated to highlight the mushroom views from several perspectives.

In this project we present *MushroomApp*, an application to identify mushrooms. The main objective is to identify a mushroom by a simple picture taken by a mobile phone. The picture is delivered to a central server. After processing the received image, the server answer with the scientific name of the mushroom and their features. You can also download and upload mushrooms and check for the correct labeling of them.

Another objective of this project is to made an initial model for the detection of mushrooms. It is intended to be an initial model. To achieve this objective, we have studied many supervised machine-learning classifiers, and decided the usage of Artificial Neural Networks (ANN). ANNs are created with the Google library TensorFlow, which in turn is one of the most widely used, due to its high efficiency. It is used not only on image detecting, but audio and text processing. We chosen TensorFlow because it is open-source and its flexibility and a large developer community have positioned it as the leading tool in the Deep Learning sector.

The data corpus is limited to the most popular edible mushrooms of Lleida, the province with the largest forests of Catalonia. *MushroomApp* can help mycologists to classify the different types of mushrooms of our environment. In addition it can be a first guide to avoid poisoning due to mushroom intake. Nowadays, many of our daily tasks are digitized, so implementing an application so that you can search for mushrooms would be very convenient.

All the uploaded images will be saved and tagged accordingly by a mycologist. This continuously re-feed the corpus used to regenerate the model. This learning process improves gradually the accuracy of the model, as it is shown in the results section. The model presented scales almost linearly, thus expectation to obtain an optimal model is so encouraging.

The main contributions are in four directions. The first one consist of making a user friendly application to help classifying mushrooms. The second is to provide a means to upload mushroom pictures from any user registered in the app. The third contribution, the most important one, is to made up a consistent data base of the most popular edible mushrooms of Lleida. Finally, providing a consistent and efficient learning model to guess local mushrooms which increase accuracy with the mushrooms images saved in the database.

II. State of the Art

There are many applications in the market. The applications providing similar features to *MushroomApp* have been analyzed. The study takes into account the most outstanding apps of the last year [4] or best commented on specialized forums [3], [5]. Table I shows the detailed description of the analyzed apps. The applications are available in *iTunes*[1] and *Google Play*[2].

[1]iTunes: https://www.apple.com/es/itunes/

[2]Google Play: https://play.google.com/store

app	Description
MushroomApp	**It recognizes local mushrooms with ANN models** **Only athorized users can tag the new images**
Setas Pro	Visualizes mushroom images and their descriptions Do not recognize mushrooms
Boletus Lite	Provides mushroom recognition based on a supervised classification algorithm
Setas Bolets	Users can upload images Users can save the location of their favorite mushrooms
Mushrooms app	Provides mushroom recognition Provides a consultation catalog
Mushroom Identify	Provides mushroom recognition Provides geolocation of mushrooms

TABLE I: Main features of related mushrooms apps.

Table II shows the app downloads and their ratings and the existence of a premium version providing access to additional information. Customers can rate your app on a scale from 1 to 5 stars. They can also write a review for your iOS and macOS apps. When a customer edits their rating or review, the most recent change will display on your App Store product page. If a customer submits a new rating or review, the existing customer review is replaced.

app	Downloads (thousands)	Average	PRO Rating
MushroomApp	**N/A**	**N/A**	**N/A**
Setas Pro - Nature Mobile	1,000 - 10,000	3.7	X
Boletus Lite - Buscador Setas	10,000 - 50,000	3.6	X
Setas Bolets - MushTool	100,000 - 500,000	3.9	-
Mushrooms app	500,000 - 1,000,000	4.3	X
Mushroom Identify	1,000,000 - 2,000,000	4.2	X

TABLE II: Mushrooms apps popularity.

III. Methods

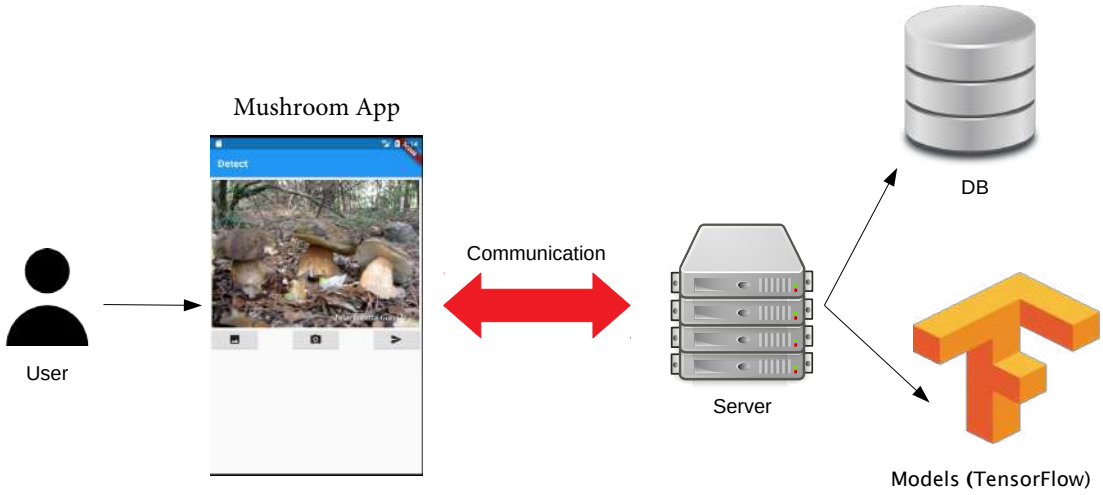

Fig. 1: *MushroomApp* framework.

Figure 1 shows the architecture of the overall framework of the overall mobile app implementation. The mobile app interacts with the server, which respond to requests made by the user through the app.

The database (DB), located on the server, contains the mushrooms images. The Efficiency design principle guided the decision to locate the DB in the server, because the app should be as light as possible. The models used for detecting mushrooms are generated by using the mushroom images stored in the DB.

A. Data Corpus

As there are many types of mushrooms, an important design decision was to mark the mushroom range of the MushroomApp-model scope. Detection precision and time used in training, testing and executing the model are the most important performance metrics taken into account.

Firstly, the model was made by a relatively small corpus, made from mushroom images obtained from the Internet and after being transformed, in order to improve its recognition, they were stored in the DB.

The model should be raised gradually by increasing the corpus. The users insert new pictures of mushrooms directly form the server or made by a (mobile) camera and upload them by means of the app. Next, an expert mycologist will tag these new pictures accordingly before being added to the corpus. Afterwards, a new mushroom ANN model is generated. This is the base principle of the learning procedure of *MushroomApp*. Thus detection accuracy increases by enlarging the number of images labeled. This is, the model improves with the corpus.

Table III shows the base starting mushroom catalog. Those are the most consumed mushrooms in our neighbouring zone of the province of Lleida, Catalonia [1], [2]. This was a design decision based on the interest on deploying the app in Catalonia to effectively test the accuracy of our proposal, and to provide it a reasonable response time. By increasing the corpus, the accuracy increases but also the response time of the system can drop drastically, so we limited the mushroom range to the edible ones of Lleida.

TABLE III: Mushrooms data base of *MushroomApp*

Genus	Species
Boletus	`Boletus aereus`
	`Boletus edulis`
	`Boletus pinophilus`
Cantharellus	`Cantharellus lutescens`
Hydnum	`Hydnum repandum`
Hygrophorus	`Hygrophorus agathosmus`
	`Hygrophorus eburneus`
	`Hygrophorus russula`
	`Hygrophorus latitabundus`
Lactarius	`Lactarius deliciosus`
	`Lactarius sanguifluus`
	`Lactarius semisanguifluus`
	`Lactarius vinosus`
Suillus	`Suillus luteus`
	`Suillus variegatus`
Tricholoma	`Tricholoma terreum`

The mushroom images have been classified according to genus and species (see Table III). The organization of the mushroom images has the following structure. There is one general folder of genus (i.e. Boletus), which contains two subfolders:

1) One with images of the specific genus (i.e. Boletus, see Fig. 2).

Fig. 2: Boletus images.

2) One with images of other genus, except the specific one (i.e. Not Boletus, see Fig. 3).

Fig. 3: Not Boletus images.

B. *MushroomApp*

Currently, there are many alternative technologies to develop applications for mobile devices. Developing the application in native code for Android and iOS should imply an extraordinary implementation cost. The *MushroomApp* application was developed with *Flutter*. The Flutter environment was chosen mainly due to 4 features it provides:

- Efficient APIs and libraries. It provide an optimal app development.
- Support for Graphical User Inerface (GUI). Flexible and expressive interface of the apps.
- Cross-platform native performance. Performance is similar to native-code apps.
- Open-source. Apps can be deployed and customized at will.

The app consists of three operations:

1) **Guessing Mushrooms:** In this function, the user will be asked to insert an image and send it to the server. When sending the image, the server will reply with the result of the request. This is, the genus and species found. Upon receiving the data, the characteristics of the mushroom are also shown (see Fig. 4).

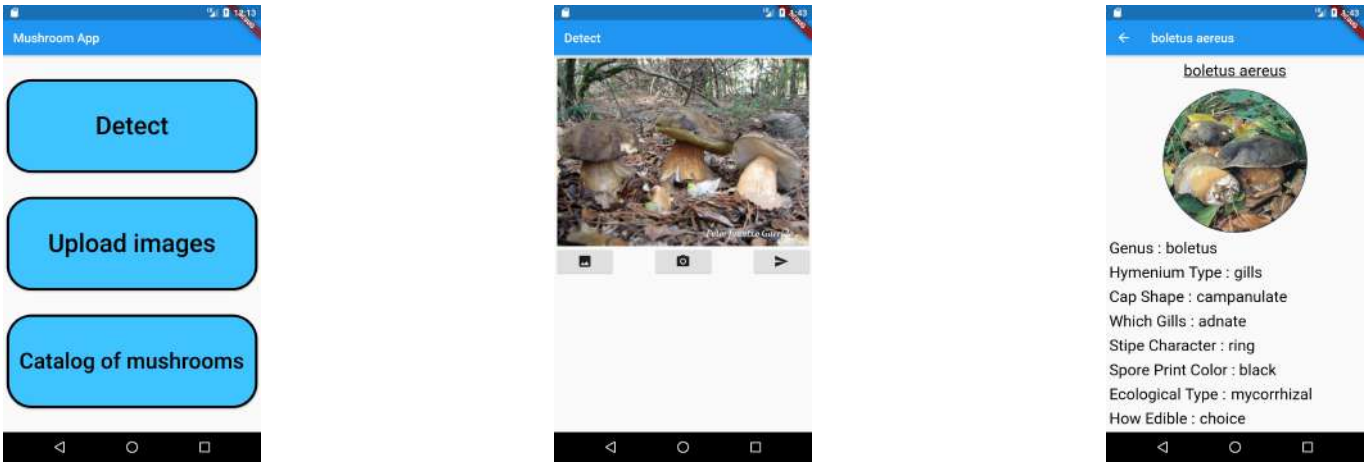

Fig. 4: *MushroomApp* Guess.

2) **Uploading images:** This an operation for administrative users only. They will be responsible for uploading images by the different allowed or predetermined criteria to create the NNA models. These images will help to increase the system learning by creating new models with more images (see Fig. 5).

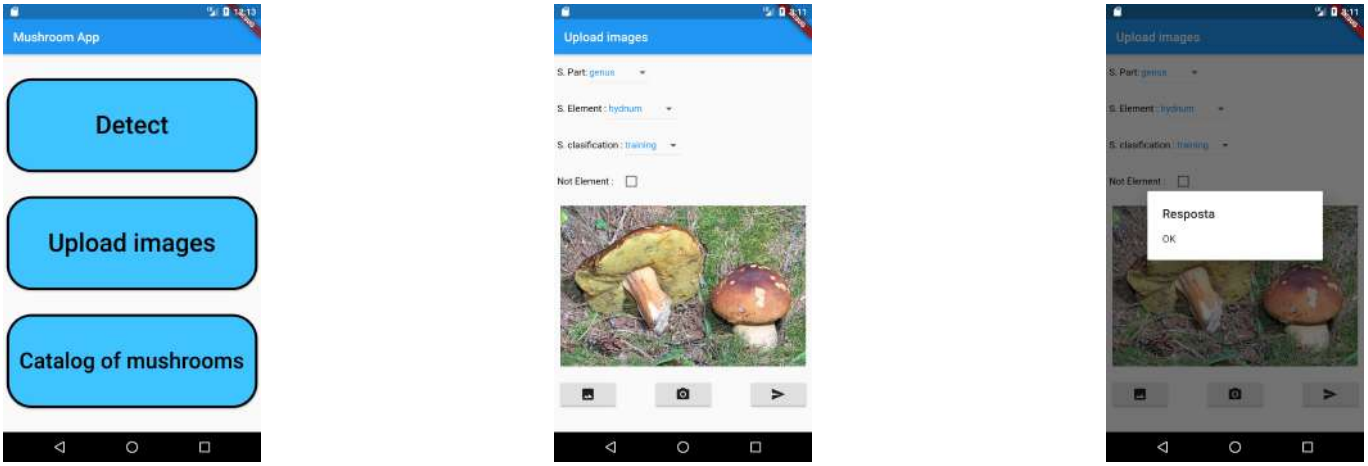

Fig. 5: *MushroomApp* Upload.

3) **Cataloging Mushrooms:** When the user clicks on the catalog, the server will send all the mushrooms registered in the DB. The user can consult the characteristics of each mushroom (see Fig. 6).

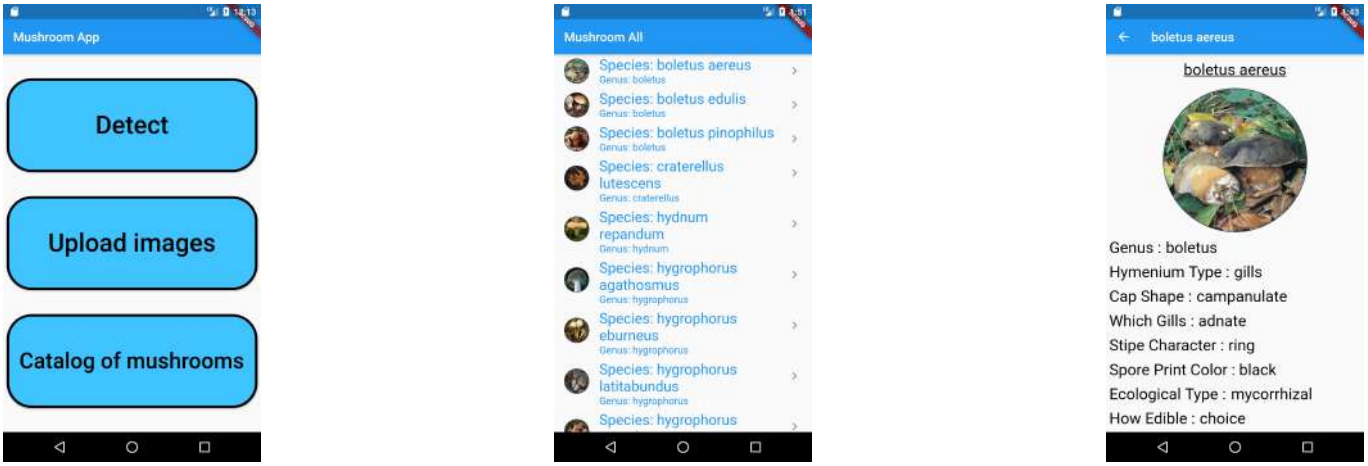

Fig. 6: *MushroomApp* Cataloging.

C. Server

The server is implemented with *Python* and the *Django* framework. Django is thought to assist development of apps because it is based on the Model View Controller (MVC).

The server provides the following functionalities:

- Registering Mushrooms along with its description and images.
- Uploading images by following different managing and storing criteria in order to creating ANN models. This functionality is equivalent to "Cataloging Mushrooms", provided by the app.
- Activating the ANNs and be able to test the detection of images.
- Consulting all requests made to the system. This shows the obtained results. The successfully results are saved in the DB and added to the model. Thus, the new ANN model is continuously learning from the expert system.

The communication with the mobile devices will be through the API developed for this purpose. The requested consultations are sent in the Json format.

D. Operation

For the creation of Artificial Neural Networks (**ANN**), we are using *Keras*. Keras is a library encoded in Python. From Keras we are calling *TensorFlow*.

Once we have defined all the models, it will be necessary to do the search for our models, to know what the seta is, that the Algorithm III-E will show at the point of the algorithm.

There are one model for each Genus **ANN** and for each Boletus, so that the system manages the following models:

- **Genus**: *Boletus, Cantharellus, Cantharellus, Hygrophorus, Lactarius, Suillus and Tricholoma.*
- **Mushroom**: *Boletus aereus, Boletus edulis, Boletus pinophilus, Hygrophorus agathosmus, Hygrophorus eburneus, Hygrophorus russula, Hygrophorus latitabundus, Lactarius deliciosus, Lactarius sanguifluus, Lactarius semisanguifluus, Lactarius vinosus, Suillus luteus and Suillus variegatus.*

The **ANN** model output is binary. For example, the model of Genus Boletus, will give the output 0 when guessing the genus Boletus and output 1 will mean that genus is not Boletus, telling us that it is another genus.

Given an image, the operation is straightforward. First, all the Genus models are applied to the image. Then, given the Genus obtained (with more accuracy), their Mushrooms models are then executed, and the best one will be the mushroom chosen.

To understand how an ANN works we must first know how an Artificial Neuron Network works.

1) Deep Learning - ANNs: Deep learning is part of a broader family of machine learning methods based on Artificial Neural Networks (ANN).

Deep learning architectures have been applied to fields including computer vision, speech recognition, natural language processing, audio recognition, social network filtering, machine translation, bioinformatics, drug design, medical image analysis, material inspection and board game programs, where they have produced results comparable to and in some cases superior to human experts [7], [8], [9]. In our case, Deep learning is applied to image recognition.

ANNs have various differences from biological brains. Specifically, neural networks tend to be static and symbolic, while the biological brain of most living organisms is dynamic (plastic) and analog [10], [11], [12].

An Artificial Neural Network (ANN) is a network of simple elements called artificial neurons, which receive inputs, change their internal state (activation) according to that input, and produce output depending on the input and activation. The image 7 corresponds to an Artificial Neuron. The inputs are x_1, x_2, x_3 and the output is y. The weights are w_1, w_2, w_3. In this case the output is a value between 0 and 1.

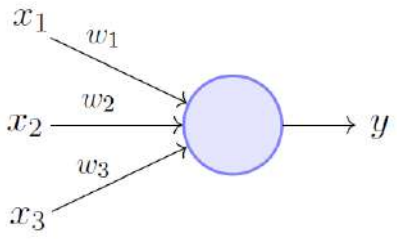

Fig. 7: Artificial Neuron.

An ANN network is formed by connecting the output of certain neurons to the input of another ones forming a directed, weighted graph. The weights as well as the functions that compute the activation can be modified by a process called learning which is governed by a learning rule [6].

The image 8 corresponds to an ANN. It is formed by three layers. The first layer is the input layer and the last is output. The intermediate layer is the hidden layer. In this image there is only one hidden layer. There is no restrictions in the number of hidden layers. However, the more hidden layers, the more

complexity of the ANN. The complexity determines the time spent in the generation of the models in the training and testing phases.

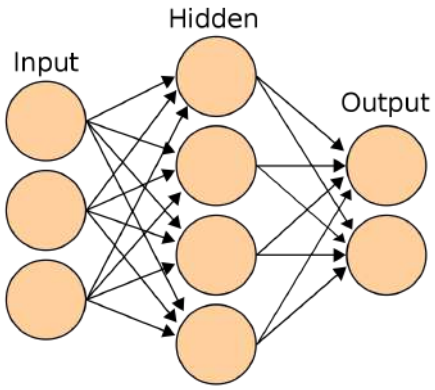

Fig. 8: Artificial Neural Network.

E. Algorithms

Algorithm 1 corresponds to the main process of the app. The input of this algorithm is the mushroom picture send by the user by means of the app. The picture can be taken by the app itself by using the photograph application contained in all the current smartphones.

In line 2, the algorithm 1 obtains all different genus of the DB. Then, the genus of the picture is obtained (line 3). Then the specific Mushroom is obtained (lines 5-6). Then, properties of the obtained mushroom are found (lines 8-9), and returned to the user (line 11).

Algorithm 1 Guessing Mushroom

```
    ObtainData: Receive File
 2: DB: Select in ElementFilteringTF: listGenus
    typeGenus = predictGenus(listGenus, file)
 4: if typeGenus ≠ Ø then
        DB: Select in Mushroom: mushroom
 6:     if mushroom > 1 then
            mushroom = predictMushroom(typeGenus, file)
 8:         if typeMushroom ≠ Ø then
                DB: Select in Mushroom: mushroom
10:             return mushroom
            else
12:             return Ø
            end if
14:     end if
        return mushroom
16: else
        return Ø
18: end if
```

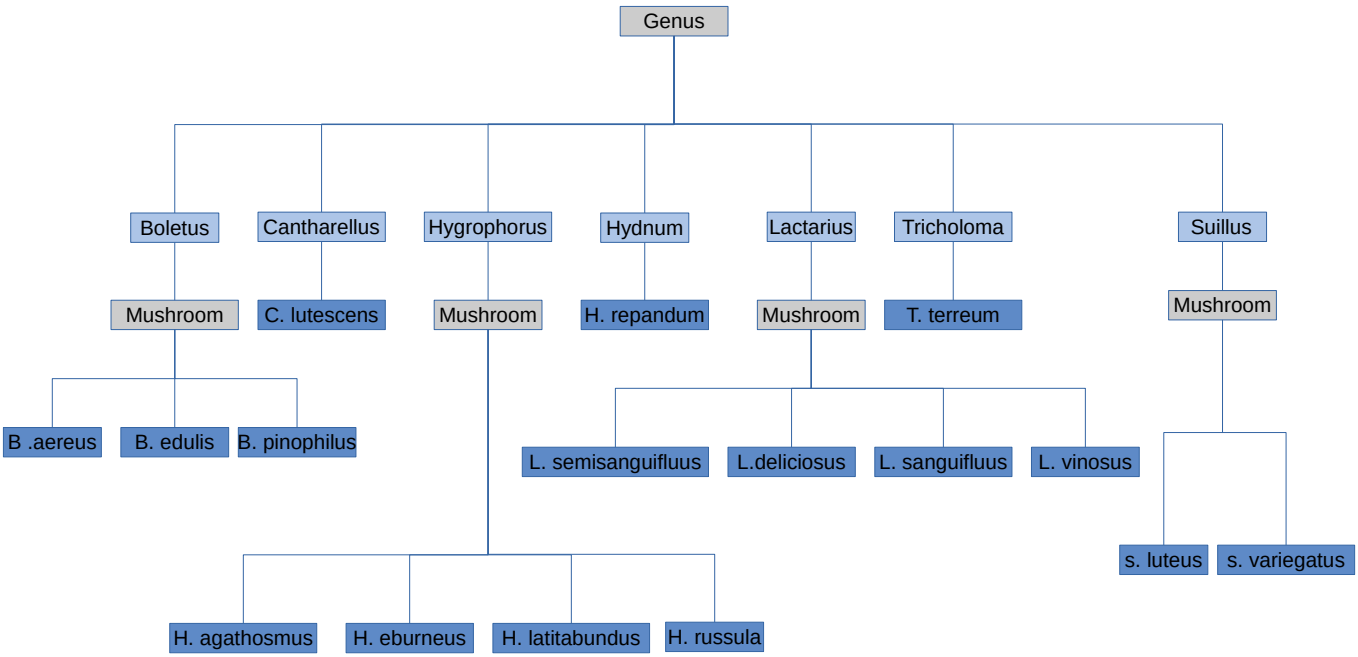

Fig. 9: Flowchart of Algorithm 1.

Fig. 9 shows the flowchart that follows the Algorithm 1. The first model to find the image will be Genus. If the result of the Genus model is Cantharellus, Hydnum or Tricholoma, the prediction process will be over since they only contain a mushrooom. In any other case, the genus mushroom models will be executed, to determine the specific mushroom.

Algorithms 2 and 3 are responsible for preparing the arguments and calling the *Predict* function, which in turn is similar for predicting genus and mushrooms. The arguments are the genus or mushroom model to be executed jointly with the genus or mushroom list and the taken picture.

Algorithm 2 Mushroom Predict Genus

predictGenus(genusList, file):
2: **return** $predictElement('genus', genusList, file)$

Algorithm 3 Predict Genus and Mushroom

predictMushroom(genus, file):
2: **DB:** Select in ElementFilteringTF: $mushroomList$
return $predictElement('mushroom', mushroomList, file)$

Algorithm 4 is responsible for predicting the genus and mushrooms. It finds across the overall list of genus or mushrooms. When finish, it returns the obtained genus of mushroom (answer 0). For each element in the list, its corresponding ANN (TensorFlow) model is called and executed for all the elements (genus or mushroom) until the prediction returns 0, meaning a match is found.

F. Data Base (DB)

The database used in the system is *MongoDb*. *Django* provides a framework to program and manage the Mongo DB. MongoDB is a document-oriented NoSQL database used for high volume data storage. It

Algorithm 4 Mushroom Predict

```
predictElement(part, element, file):
for each e ∈ element do
    LoadModels: We load TensorFlow models.
    PredictImage: We consult the file in the model.
    answer = Prediction result.
    if answer == 0 then
        return e
        break
    end if
end for each
```

falls under the category of a NoSQL database. For batch processing of data and aggregation operations, MapReduce can be used. MapReduce is nothing but an associated implementation for processing and generating big data sets in parallel. The major and very common problem with a growing web application is scaling. To overcome this, MongoDB, distributes data across multiple machines. By means of MapReduce, huge amount of data can be processed in parallel then by accessing the machines simultaneously.

We next present the MongoDB tables making up the DB. Their forming fields as well as a desciption on them is performed.

Table IV contain the name of the filters used for the prediction. As only two kind of predictions are implemented (genus and mushroom),it is made of only two registers.

Atribute	**Description**
Part	Genus-model Name

TABLE IV: Table PartFilteringTF.

Table V contains the classification of Genus and Mushrooms. The forming elements are identified by PartFyrilteringTF as a foreign key and then the name of the element. For example, possible records are "Boletus" that refers to "Genus" or "Boletus aereus" that refers to "Mushroom".

Atribute	**Description**
Part	ForeignKey(PartFilteringTF)
Element	Mushroom-model Name

TABLE V: Table ElementFilteringTF.

Table VI shows the fields and they corresponding types of each register, used in the identification phase. They are used in making the different models for each genus and mushroom.

The fields Part and Element are the primary keys. Tipus is a binary variable that informs that this register will be used in the training, in the classification process. NotElement is a checking boolean variable, used to introduce error images, or bad classified. Photo contains the Url of the image. Only Url are saved, because a DB containing the images itself, should be so huge.

Atribute	Description
Part	ForeignKey(PartFilteringTF)
Element	ForeignKey(ElementFilteringTF)
Tipus	Binay field containing “Training” or “Validation”)
NotElement	Boolean variable. A value equals to 1 means a bad classified element. When training this option is set.
Photo	Url photo of the picture

TABLE VI: Table ElementImage.

Table VI shows the fields and they corresponding types of each register, used to record the mushrooms available in the system. The field Name is species type. Genus is the mushroom genus. HymeniumType is a select box with the hymenial surface type. CapShape is a checkbox indicating cap shape. EcologicalType is a select box with ecologycal annotations. HowEdible is a select box indicating edibility. Description contains additional information of the mushroom.

Atribute	Description
Name	The species type
Genus	The genus type
HymeniumType	Type of spore-bearing surface: gills, pores, etc.
CapShape	Descriptor of the general shape of the cap
whichGills	Descriptor of how the hymenium attaches to the stem. Applies even to ridged, toothed and pored species, despite parameter name
StipeCharacter	Indicates if a universal or partial veil is present
SporePrintColor	Color of the spore
EcologicalType	Indicates how the mushroom obtains nutrients
HowEdible	Indicates whether the mushroom is edible or poisonous
Description	Mushroom description

TABLE VII: Table Mushrooms.

Table VIII shows the fields and they corresponding types of each register, used to record the results of mushroom predictions. The field Mushroom is a string that the name of the mushroom that has been obtained from the prediction reported. Photo contains the Url of the image. Only Url are saved because a DB containing the images itself, should be so huge.

Attribute	Description
Mushroom	Result prediction
Photo	Url photo in the server

TABLE VIII: Table PredictImage.

IV. Results

In this section we have evaluated the prediction of the models (TensorFlow). We are going to evaluate it with the results of the Precision, Recall and the F-score, based on the following possibilities .

- *True positive (tp):* mushroom image in the DB correctly identified.
- *False positive (fp):* mushroom image not in the DB incorrectly identified.
- *True negative (tn):* mushroom image not in the DB correctly detected as unknown.
- *False negative (fn):* mushroom image in the DB incorrectly detected as unknown.

Precision, Recall and F-score are defined by the formulas 1, 2 and 3 respectively.

$$Precision = \frac{tp}{tp + fp} \tag{1}$$

$$Recall = \frac{tp}{tp + fn} \tag{2}$$

$$F - score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{3}$$

A. *Testing Genus*

The results of the tests performed are shown below. The efficiency of our proposal of using an Artificial Neural Network (ANN) will be tested by comparing two models made with a corpus of 10,000 and another of 27,000 images. This is intended to demonstrate that the efficiency of the models grows with the size of the dataset.

According to Fig. 10, the obtained F-score with 27,000 images was 0.68. With fewer images (10,000), the F-score decreased until 0.5. Therefore, it is shown that the performance depends on the size of the corpus.

However, an open question should be to tune the corpus used. Perhaps better image filtering, transforming, or annotation can increase even more the performance. This is an issue to be taken into account in future work.

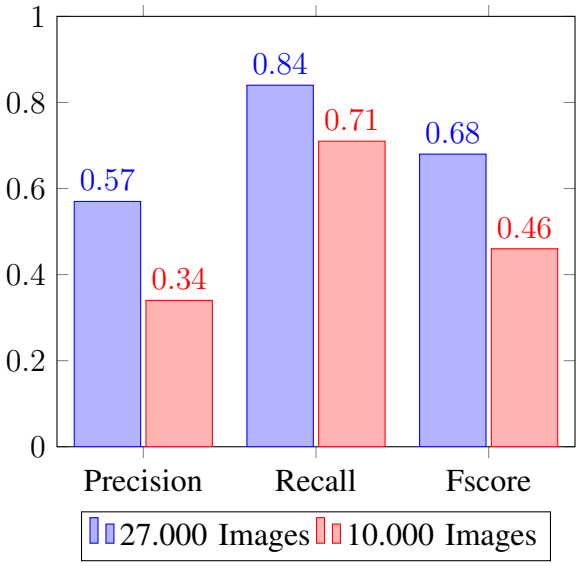

Fig. 10: *MushroomApp* results for genus.

B. *Testing Mushroom*

We have divided the performance evaluation of mushroom prediction into 4 parts. One for each of the 4 types of mushrooms Boletus, Hygrophorus, Lactarius and Suillus. These mushrooms have been selected because they have the largest number of images.

Figures 11, 12, 12 and 14 show the result of the tests. In this case, the tested ANN models have been generated with a dataset of 7,000 and 5,0000 images.

In all the cases, the F-score is below 0.5. It should be noted that the best results have been obtained with datasets of 7,000 images. Therefore, it can be assured that the prediction accuracy increases with the size of the dataset.

We can affirm that to have good results by using ANN model, larger image corpus must be collected.

As was commented in the genus case, an open question should be to better tune the corpus used.

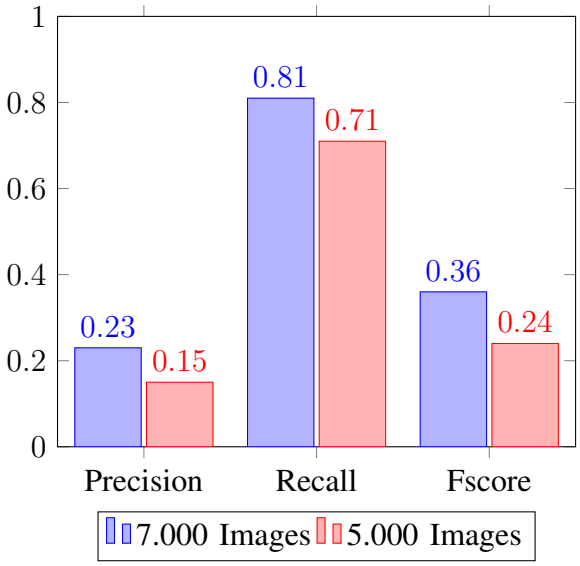

Fig. 11: *MushroomApp* results for mushroom - Boletus.

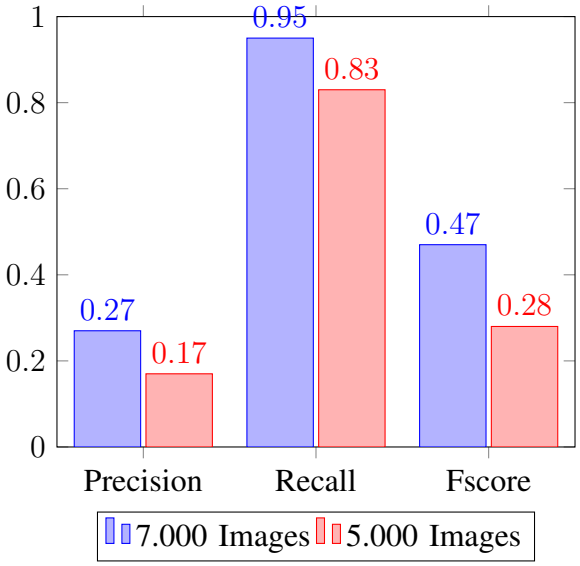

Fig. 12: *MushroomApp* results for mushroom - Hygrophorus.

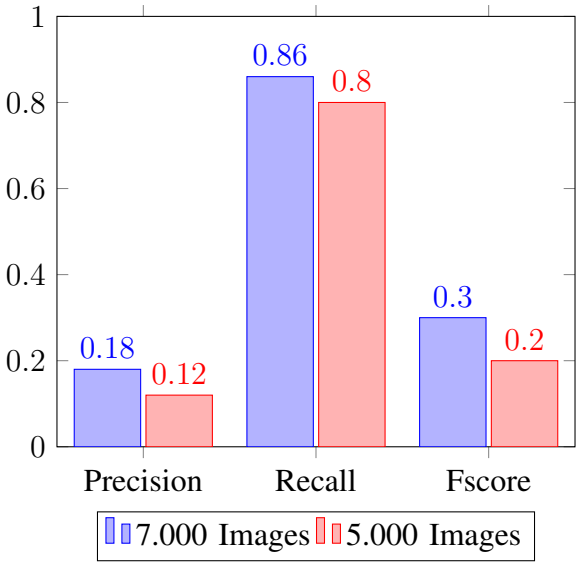

Fig. 13: *MushroomApp* results for mushroom - Lactarius.

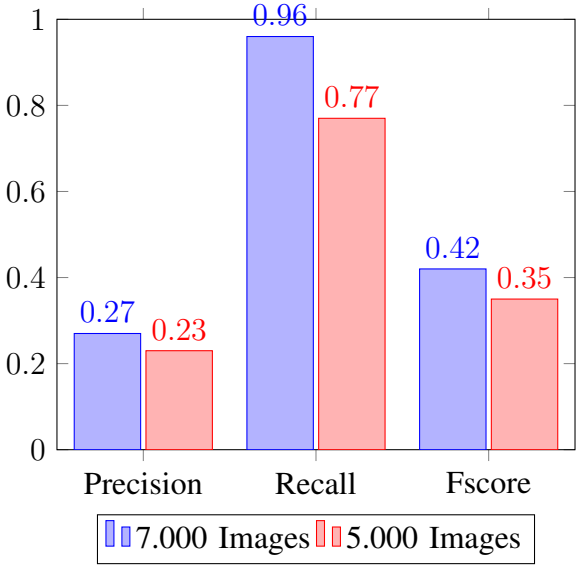

Fig. 14: *MushroomApp* results mushroom - Suillus.

C. Creation time

The creation time of an ANN is indifferent to the size of dataset used (10,000 or 27,000 images). Even with 10,000 images, it has sometimes taken longer because the resolution of the ANN is not determined linearly in the search for the solution due to the lack of images. This is because the ANN cannot trace paths in the search for the optimum.

But if the number of steps and cycles is high, it can take longer. If we have many hidden layers it may also take longer.

In this sense one would expect to perform much more experimentation to find the most efficient corpus in terms of accuracy and response time training and executing the ANN models.

V. Conclusions and future

We developed and initial App prototype to guess comestible mushroom of Catalonia, called *MushroomApp*.

The most remarkable conclusion is that to date, we have not enough pictures of mushrooms in order to have good performance detecting them. It should be necessary to compile and tune more pictures to provide the application with acceptable performance of the ANN models. However, the app is provided with the appropriate means to enlarge the repository.

The future trend should be in the following directions:

- Improving the corpus. Both annotation and size.
- Adding new models. Found with ANN or another kind of Deep Learning technology.
- Add Poisonous Mushrooms.
- Adding a location field.
- Capability for uploading multiple mushroom images at the same time.

REFERENCES

[1] de-Miguel, S., Bonet, J.A., Pukkala, T., Martínez de Aragón, J., 2014. Impact of forest management intensity on landscape-level mushroom productivity: a regional modelbased scenario analysis.

[2] Edible mushrooms most commonly found in Catalonia, https://web.gencat.cat/en/actualitat/reportatges/temporada-de-bolets/bolets/bolets-comestibles/

[3] The Best Apps For Mushroom Identification, https://freshcapmushrooms.com/learn/mushroom-identification-app/

[4] The most outstanding of the last year, https://lacasadelassetas.com/blog/las-mejores-aplicaciones-para-identificar-setas/

[5] App with quite successful result to go out to look for mushrooms, https://www.eldiario.es/consumoclaro/consumo_digital/aplicaciones-movil-buscar-setas_0_559244457.html

[6] Zell, Andreas (1994). "chapter 5.2". Simulation Neuronaler Netze [Simulation of Neural Networks] (in German) (1st ed.). Addison-Wesley.

[7] Ciresan, Dan; Meier, U.; Schmidhuber, J. (June 2012). "Multi-column deep neural networks for image classification". 2012 IEEE Conference on Computer Vision and Pattern Recognition: 3642–3649.

[8] Krizhevsky, Alex; Sutskever, Ilya; Hinton, Geoffry (2012). "ImageNet Classification with Deep Convolutional Neural Networks

[9] "Google's AlphaGo AI wins three-match series against the world's best Go player". TechCrunch. 25 May 2017.

[10] Marblestone, Adam H.; Wayne, Greg; Kording, Konrad P. (2016). "Toward an Integration of Deep Learning and Neuroscience"

[11] Olshausen, B. A. (1996). "Emergence of simple-cell receptive field properties by learning a sparse code for natural images". Nature. 381 (6583): 607–609

[12] Bengio, Yoshua; Lee, Dong-Hyun; Bornschein, Jorg; Mesnard, Thomas; Lin, Zhouhan (2015-02-13). "Towards Biologically Plausible Deep Learning